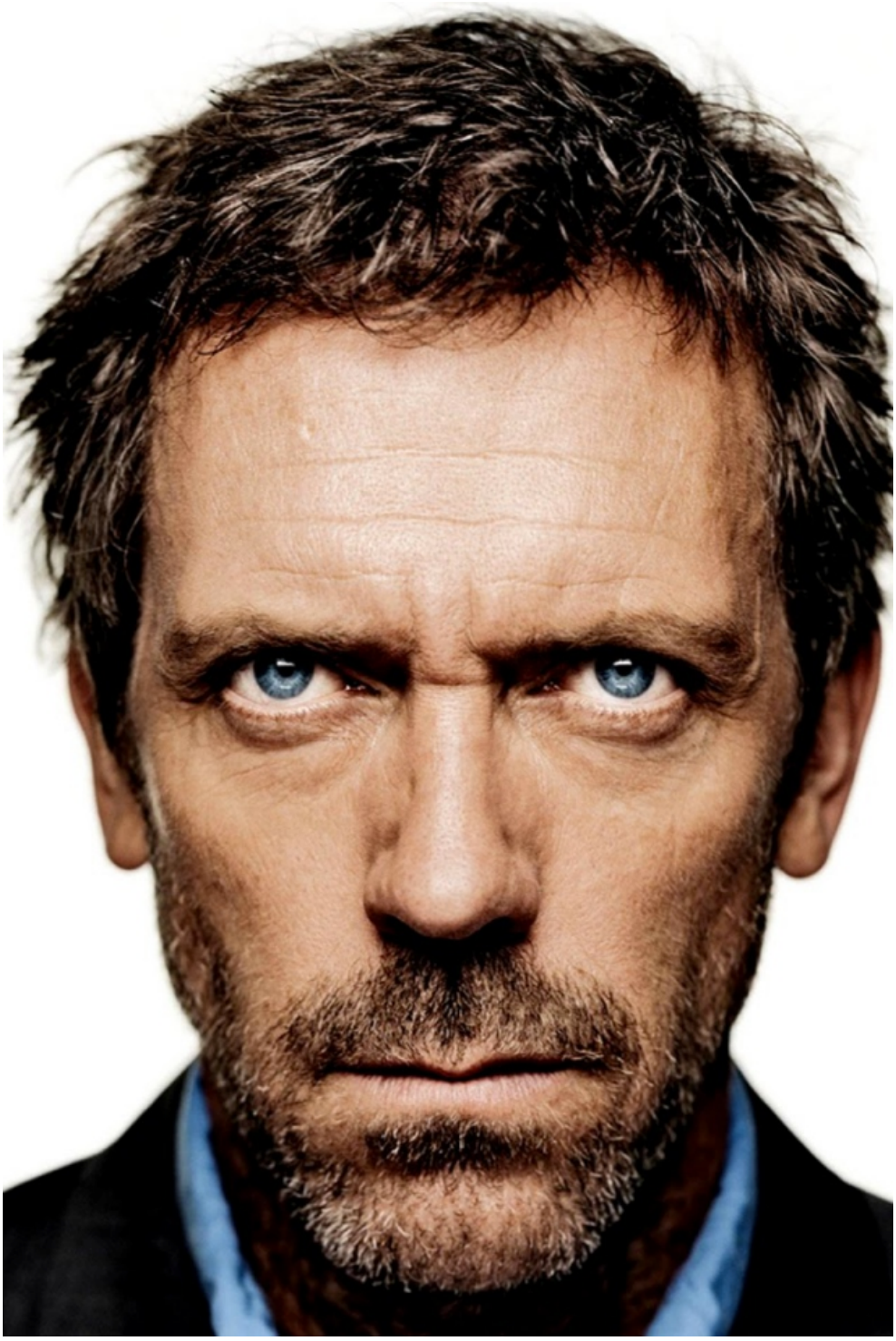
TheHouse

By David Mayo and Patrick McCabe



Introduction

Before the Pokerbots competition began we knew only the very basics about how to play poker. We decided that since neither of us had any professional poker playing experience we should delegate the responsibility of figuring out the correct strategy to exploit our opponents to our AI. By this we mean our AI player should learn from how the other players at the table play poker and adjust its strategy to exploit them without any specific strategy hardcoded into the bot.

We began our AI endeavor by first creating a tight aggressive player that completely ignores what the other players at the table are doing. This bot was code named “PatBot”.

PatBot

Our first generation poker bot was internally known as “PatBot”. The goal with PatBot was to implement some of the fundamentals of poker that we had recently read about. We wanted a bot that would, at the very least, win a few thousand dollars in the casino. PatBot’s strategy is a combination of a pre-flop and post-flop strategy. Each sub-strategy generates and returns an ActionProbability object, which randomly chooses an action (e.g. bet, call, fold, etc.) based on a given probability distribution. If the randomly chosen action requires an amount (e.g. raise $15), then PatBot calculates an amount within the minimum and maximum allowed value that is proportional to our equity.

PatBot was written before we were able to setup the provided equity calculator. In order to calculate equity quickly, PatBot uses a monte-carlo equity calculator that we wrote ourselves. This calculator only calculated our equity versus a single random opponent.

Pre-flop:

PatBot’s pre-flop strategy relies heavily on a table that ranks every starting hand (ignoring suit isomorphism) according to their equity-squared value. The higher our hole cards rank in this table the more likely PatBot is to raise pre-flop. The effect of using equity-squared instead of the standard all-in equity is to increase the value of hands that can make strong draws (i.e. straights or flushes).

PatBot is a positionally aware poker player, which means he adapts to the advantages and disadvantages that his position at the table provides him. For example, PatBot plays more hands when on the button than he does as the big blind. This is because the button has the post-flop advantage of acting last whereas the big blind acts first post-flop.

PatBot also has a few fundamental pre-flop aggressive tendencies. PatBot tries to avoid limping in when on the button - instead he likely chooses the more aggressive action of raising. As the big blind, PatBot has a high probability of raising limpers to try to steal the pot.

Post-flop:

PatBot's post-flop strategy is broken into two parts. The first part controls our strategy when the pot has yet to be opened, and the second part controls our strategy once the pot has been opened. If the pot has not been opened, meaning everyone before us has checked, then PatBot's post-flop strategy returns an ActionProbability that has a probability of raising equal to our equity and a probability of checking equal to 1-equity. This has the effect of raising our good hands and checking our bad ones. This also creates some possibility of bluffing and check-raising.

If the pot has been opened then we face the choice of calling, raising, or folding. PatBot calculates the pot-odds in order to make this decision. The pot-odds are calculated as callAmount / (callAmount + potSize). This formula tells us the minimum equity necessary to make a positive executive value call. If our equity is greater than the pot-odds then our post-flop strategy returns an ActionProbability with P(call)=0.9 and P(raise)=0.1. Otherwise, we return an ActionProbability with P(fold)=0.9, P(call)=0.05, and P(raise)=0.05. These probabilities lead to a fairly passive poker player.

Conclusions:

After creating PatBot and testing him in the casino we were surprised to find that although his stats seemed consistent with what we thought was good ABC poker he was only able to achieve a marginally positive bankroll. We believed that this was caused by PatBot not being aggressive enough and not trying to exploit the weaknesses of the other bots. Since PatBot's strategy was too rigid and predictable he was prone to exploitation by other bots.

In order to learn a little bit about our opponents strategies we printed out a number of different statistics about our opponents to the casino dump files. When we manually analyzed these statistics we found that our opponents were being extremely aggressive. Since their statistics seemed to deviate significantly from what we considered to be good poker we realized that a more exploitative strategy could be extremely effective.

StatBot

To address the pitfalls of PatBot, we created a new bot with opponent modeling in mind - StatBot. As his name implies, StatBot was built to collect, interpret, and act on his opponents statistics. The full list of statistics collected by StatBot are below, but the main ones he uses are VPIP (voluntarily put money in pot) and PFR (pre-flop raise). The remaining statistics are mostly used by us when manually reviewing our performance and the performance of other bots in the casino.

StatBot uses these statistics to put his opponents on a hand range. For example, if his opponent in the small blind simply called the big blind during pre-flop (this is a VPIP action) then we assign that opponent a hand range equal to his VPIP% for the small blind position. So if the opponent has a VPIP of 25%, we assign him a range of the top 25% of starting hands (this is done with a look up table). Had the opponent raised then we would use his PFR% instead. This range assignment is done this way because we assume that, for example, if a player raises 6% of the time then he is probably only raising with the top 6% of hands. This hand range is then used to more accurately evaluate our equity.

Like PatBot, StatBot's strategy is broken up into a pre-flop and post-flop strategy. Both of these sub-strategies return ActionProbabilities just like in PatBot. Unlike PatBot, the dollar amount for an action like raise or bet is chosen to be proportional to equity*pot / (1-equity). This gives us an amount to bet/raise with which has a positive expected value if we were to go to showdown immediately.

Statistics:

StatBot calculates and stores many statistics on each opponent he faces. If we are facing a new opponent then the statistics are initialized with values that are representative of what we consider a decent poker player. The values are then updated using an exponential moving average. Between matches, all of these statistics are stored as percentages precise to two decimal places. The statistics are reloaded when we face the opponent again in a later match and are continually updated.

VPIP - voluntarily put money in pot : This is the percentage of times a player puts money in the pot during pre-flop (ignoring blinds). This statistic is tracked for each position (button, small blind, and big blind)

PFR - pre-flop raise: This is the percentage of times a player raises during pre-flop when they have the opportunity to do so. This statistic is also tracked for each position.

WTSD - went to showdown. This is the percentage of times a player goes to showdown given they saw the flop.
W$SD - when money at showdown. This is the percentage of times a player ties or wins a pot at showdown.
W$WSF - when money when saw flop. This is the percentage of times a player ties or wins a pot given that they saw the flop.
W$WST - when money when saw turn. This is the percentage of times a player ties or wins a pot given that they saw the turn.
W$WSR - when money when saw river. This is the percentage of times a player ties or wins a pot given that they saw the river.
FPFR - fold to pre-flop raise. This is the percentage of times a player folds to a pre-flop raise.
CBET - continuation bet. This is the percentage of a times a player makes a bet on the flop given that they were the pre-flop raiser.

Pre-flop:

StatBot's pre-flop strategy takes into account his stack size and his opponent's actions through the hand-range/equity calculation. With a healthy sized stack (50 big blinds), the pre-flop strategy returns an ActionProbability with a high probability of raising when our equity is high, a high probability of calling with mid-range equity, and a high probability of folding with low equity.

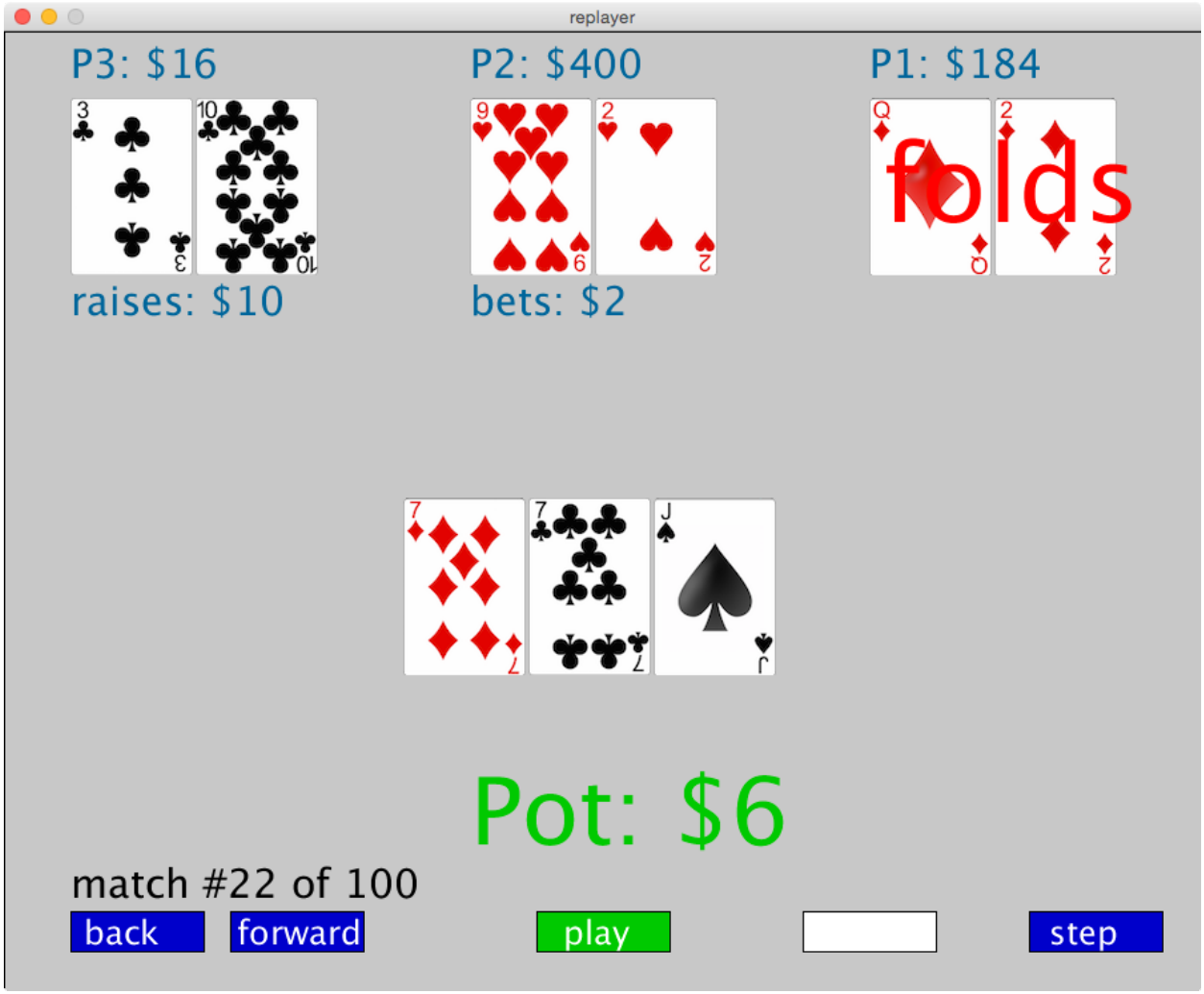
As our stack size decreases Statbot becomes tighter and more aggressive with his play - in hopes of building his stack back up to a healthy size. StatBot will begin to play fewer hands, but raise heavily with the ones that he does play. This type of play continues through into post-flop.

Post-flop:

StatBot's post-flop strategy also takes into account his stack size. With a healthy stack, StatBot's post-flop strategy is a lot like PatBot's. The only difference is that he bets more often when he has really high equity (>80%). When his stack size falls below the "healthy" threshold he ignores pot-odds and simply bets/raises his good hands and folds his bad ones. Once our stack is this low we cannot afford to make complicated plays like check-raises nor can we afford to be scared away because of bad pot-odds. Instead, we hope to do the scaring by being incredibly aggressive.

Testing and Data Analysis

After the first casino we quickly realized that we needed a way to analyze the hand histories from our matches, which are provided to us as text files. We wanted to create a human friendly interface that would allow us to replay interesting hands and discover areas for improvement. So we took a few hours to write a graphical hand replayer (seen below). This hand replayer allows us to replay entire matches and step through individual hands. It displays the cards and actions of every player.



Hand history replayer screenshot

In order to determine if the changes that we made to our bot were making it better or worse we wrote scripts to test it. Our testing procedure was typically to pit our new bot against PatBot and a random bot or pit our new bot against a previous iteration of StatBot and a random bot in 10 triplicate matches. These test were run automatically using bash scripts and then we analyzed win percentages in the matches using a python script.

We also utilized the casino dump files to output the statistics that we were collecting on ourselves and on our opponents. This allowed us to see how we were responding to an opponents strategy and manually judge if our response was correct.

727445781
VPIP: 0.6503321231281762
PFR: 0.03597695389473327
AF: 0.24433656957928804
WTSD: 0.3742541528918869
W$SD: 0.24613044318434638
R3B: 0.07781731130311677
CBet: 0.4618600974645
W$WSF: 0.4159915529398317
W$WST: 0.40208838928038937
W$WSR: 0.5017429662160177
VPIP FIRST 1.0010555906309526
VPIP MIDDLE 0.4356872659537705
VPIP LAST 0.21520577442314895
PFR FIRST 0.053564671565321784
PFR MIDDLE
0.004265366565247818
PFR LAST 0.023080905334761745
Fold FIRST 0.6
Fold MIDDLE 0.7241092114119405
Fold LAST 0.5764631408875602

Dump file stats for one player